Clicker Qs for November 19
What will this code evaluate to in standard Scheme?

(define zero! (lambda (x) (set! x 0)))
(define x 34)
(zero! x)

A. It is an error.
B. 34
C. 0
D. It doesn't return anything because set! doesn't.
Answer B: 34 (zero! changes its parameter, not its argument)
What about this code?
(define myDelay (lambda (exp) (lambda () exp)))
(define myForce (lambda (p) (p)))
(define x 5)
(define promise (myDelay (+ x 3)))
(myForce promise)

A. It gives an error.
B. 3
C. 8
D. It returns (+ x 3) as an unevaluated expression
Answer C: 8
What about this code? It is the same as the previous example with (define x 5) moved down two lines.

(define myDelay (lambda (exp) (lambda () exp)))
(define myForce (lambda (p) (p)))
(define promise (myDelay (+ x 3)))
(define x 5)
(myForce promise)

A. It gives an error.
B. 3
C. 8
D. It returns (+ x 3) as an unevaluated expression
Answer A: This time it gives an error because myDelay can't evaluate its argument (+ x 3).
Foobar has some Scheme value.
(define Barfoo (cons 1 Foobar))
What is (car Barfoo)?

A. It depends on the value of Foobar
B. It is an error if Foobar isn't a list.
C. 1
D. null
Answer C: 1
(define ones (cons 1 ones))
What is (car ones)?

A. It is an error
B. 1
C. It would be 1 if the definition of ones ever terminated.
D. Ones
Answer C: It would be 1 but the definition of ones terminated.